*Complessi*

Un complesso è un insieme graduato S={Si}i∈I, cioè una famiglia di insiemi, indicizzati in questo articolo su I = {0, 1, 2, 3}. Noi usiamo due tipi diversi ma intrecciati di complessi, e specificamente complessi di cellule e complessi di catene. Le loro definizioni e alcuni concetti correlati sono forniti in questa sezione. Le lettere greche sono usate per le celle di una partizione spaziale, e lettere romane per catene di celle, codificate come interi (senza) segno o matrici sparse di interi (senza) segno.

Definizione 2.1.1 (d-varietà). Una varietà è uno spazio topologico che assomiglia a uno spazio piatto localmente, cioè vicino a ogni punto. Ogni punto di una varietà d -dimensionale ha un intorno omeomorfo a Ed , lo spazio euclideo di dimensione d. Quindi, questo oggetto geometrico è spesso indicato come d-varietà. Definizione 2.1.2 (Cella). Una p-cella σ è una p-varietà (0 ≤ p ≤ d) che è lineare a tratti, connessa, possibilmente non convessa, e non necessariamente contrattile .

Nota. Ci occupiamo qui di celle Piecewise-Linear (PL) di dimensione 0, 1, 2 e 3, rispettivamente. Si dovrebbe notare che Le celle a 2 e 3 possono contenere fori, pur rimanendo collegate. In altre parole, le cellule sono p-poliedri, cioè segmenti, poligoni e poliedri immersi in uno spazio bi o tridimensionale. Anche se sono spesso convesse, le cellule in una decomposizione poliedrica di uno spazio non sono necessariamente convesse.

Definizione 2.1.3 (Complesso cellulare). Un p-complesso cellulare è un insieme finito di celle che hanno al massimo dimensione p, insieme con tutte le loro facce r-dimensionali (0 ≤ r ≤ p). Una faccia è un elemento del confine PL di una cella, che soddisfa la condizione di compatibilità di confine . Due p-celle α e β sono compatibili con i limiti quando la loro intersezione di punti contiene le stesse r-facce (0 ≤ r ≤ p) di α e β. Un p-complesso cellulare è regolare quando ogni r-cella (0 ≤ r ≤ p) è la faccia di una p-cella. Definizione 2.1.4 (Scheletro). L’ s-scheletro di un p-complesso Λp (s ≤ p) è l'insieme Λs ⊆ Λp di tutte le r-celle (r ≤ s) di Λp . Ogni scheletro di un complesso regolare è un sottocomplesso regolare.

Definizione 2.1.5 (Spazio di supporto). Lo spazio di supporto |Λ| di un complesso cellulare è l'unione puntuale delle sue cellule.

Osservazione (rappresentazione geometrica). La rappresentazione LAR di un PL complesso Λp nello spazio euclideo Ed(p ≤ d) è data da una mappa di immersione μ ∶ Λ0→ Ed , e dagli insiemi discreti U0∶= χ(Λ0), e Ur ∶= χ (Λr – Λr-1), con 1 ≤ r ≤ p, dove χ ∶ Λ → P (Λ0 ) è la funzione caratteristica da celle a sottoinsiemi di 0-celle , che collega ogni cella al sottoinsieme dei suoi “vertici”.

Osservazione (Rappresentazione minima). La rappresentazione cellulare data sopra è in realtà ridondante, poiché le mappe FV ("facce per vertici") possono essere generate da V ed EV ("bordi per vertici") utilizzando i metodi forniti in Paoluzzi et al. (2017b), che sarà riassunto nella prima metà della Sezione 3. In altre parole, dato l'incorporamento PL di un piano grafico in un piano, il 2-complesso cellulare (cioè il complemento del grafo) della sua partizione piana è completamente specificato.

Osservazione (Matrici caratteristiche). Data la rappresentazione geometrica di un p -complesso cellulare, la topologia delle s-celle ordinate (0 ≤ s ≤ p), indicato con λi ∈ Us , è completamente rappresentata dalle matrici binarie sparse Ks= [χ (λi )], dove ciascuna riga binaria fornisce l'immagine della funzione caratteristica della s-cella rispetto all'insieme di 0-celle.

Algoritmo 1 (Matrici caratteristiche). Sono usate per indicare le cellule di un complesso cellulare come vettori binari, cioè come righe di matrici binarie. Il frammento di codice seguente calcola la matrice caratteristica per un insieme CV di celle, rappresentato da vertici. La funzione K restituisce la matrice Julia di tipo Sparse Arrays fornendo la matrice caratteristica Kr dando come input un array CV che specifica ogni r-cella come array di indici di vertice. L'incorporamento di cellule, cioè la mappa affine che li trova in Ed , sarà specificato da un array V d× n, dove n è il numero di 0-celle 0.

1. function K(CV)

2. I = vcat( [ [k for h in CV[k]] per k=1:lenght(CV) ]...)

3 J = vcat(CV...)

4 X = Int8[1 per k=1:lenght6(I)]

5 return SparseArrays.sparse(I,J,X)

6 end

Definizione 2.1.6 (matrice sparsa CSC). Il formato della colonna sparsa compressa (CSC) rappresenta una matrice sparsa M di tre array (unidimensionali). In Julia, CSC è il formato di archiviazione preferito (in realtà unico, ormai) per matrici sparse. La sparsa M viene memorizzata come una struttura contenente: numero di righe, numero di colonne, puntatori colonna, riga indici di valori memorizzati e valori memorizzati, tipicamente diversi da zero.

Dati 2.1.2 (Matrici caratteristiche). Le matrici binarie sparse K1 = K(EV) e K2 = K(FV) sono generate da dati del 2-complesso 2.1.1. Qui mostriamo le corrispondenti matrici dense per motivi di leggibilità. Naturalmente, lo spazio di archiviazione di una matrice sparsa è lineare con il numero totale di elementi diversi da zero e la scarsità, definita come 0 ≤ 1 – nz∕n2 ≤ 1, cresce con il numero n di celle, ovvero la lunghezza dell'array EV.

Definizione 2.1.7 (Catena) Una p-catena può essere vista, con qualche abuso di linguaggio, come un insieme di p-cellule. In questo modo possiamo scrivere Cp =P (Up) per lo spazio delle catene e Up= Λp – Λp-1  per l'insieme delle catene unitarie (0 ≤ p ≤ d).

Definizione 2.1.8 (Spazio delle catene). L'insieme C = ⊕ Cp , somma diretta di spazi di catena, può dare la struttura di un spazio vettoriale graduato (vedi Definizione 2.1.10 e Arnold, 2018, pagg. 11-12) definendo somme di catene con la stessa dimensione, e prodotti per i tempi scalari in un campo, con le solite proprietà.

Definizione 2.1.9 (Basi dello spazio di catena). Come uno spazio lineare, ogni insieme contiene un insieme naturale di generatori di irriducibili. La base naturale Up ⊂ Cp è l'insieme delle catene indipendenti (o elementari) up ∈ Cp, dato da elementi singleton. Di conseguenza, ogni catena c ∈ Cp può essere scritta come una combinazione lineare della base con elementi di campo, ed è generato in modo univoco. Una volta fissata la base, la rappresentazione della coordinata senza segno di ogni { λk } = uk ∈ Cp è unica, ed è un array binario con un solo elemento diverso da zero in posizione e tutti gli altri elementi 0. La sequenza ordinata di scalari possono essere estratti sia da {0, 1} (rappresentazione senza segno) sia da {−1, 0,+1} (rappresentazione con segno). Con un abuso di linguaggio, spesso chiamiamo p-cellule i generatori indipendenti di Cp, ovvero gli elementi di Up .

Definizione 2.1.10 (Spazio vettoriale graduato). Uno spazio vettoriale graduato è uno spazio vettoriale V espresso come somma diretta di spazi Vp indicizzato per interi in [0, d] ∶= {p ∈ ℕ | 0 ≤ p ≤d}: = ⊕p=0d Vp . Una mappa lineare f ∶ V→W tra spazi vettoriali graduati è una mappa graduata di grado k se f(Vp ) ⊂ Wp+k per ogni p .

Definizione 2.1.11 (Complesso di catena). Un complesso di catena è uno spazio vettoriale graduato (fornito con una mappa lineare graduata) δ∶ V→ V di grado −1 detto operatore di confine, che soddisfa δ 2= 0. In altre parole, un complesso di catena è una sequenza di spazi vettoriali Cp e mappe lineari δp ∶ Cp → Cp-1, tali che δp+1 ◦ δp = 0.

Definizione 2.1.12 (complesso di co-catena). Un complesso di co-catene è uno spazio vettoriale graduato V fornito con una mappa lineare graduata δp∶ V→ V di grado +1 detto operatore di co-confine, che soddisfa δ2= 0. Vale a dire, un complesso di co-catene è una successione di spazi vettoriali Cp e mappe lineari δp∶ Cp→ Cp+1 tale che δp+1 ◦δp = 0.

Proprietà 2.1.1 (Dualità dei complessi di catena). Qualsiasi spazio di catena, essendo lineare, è associato a un unico spazio duale di co-catene. Una mappa lineare L∶ W→ Vtra spazi lineari induce una doppia mappa L\*∶ W\*→ V\* tra il loro spazi duali. Se (e solo se) lo spazio della catena primaria ha dimensione finita, come nel nostro caso, allora il suo spazio duale della co-catena ha la stessa dimensione, ed è quindi linearmente isomorfa al primordiale. Inoltre, l'operatore co-confine δk è il duale dell'operatore di confine δk+1:

(δkw)g =w(δk+1 g), per ogni w ∈ Ck , g∈Ck+1.

Esistono infiniti isomorfismi lineari tra uno spazio lineare a dimensione finita e il suo duale (e selezionare uno di essi equivale a dotare lo spazio primordiale di una struttura metrica). Ciascuno di questi isomorfismi è prodotto identificando per elemento una base dello spazio primordiale con una base del suo duale. Visto che siamo solo interessati alle proprietà topologiche, adottiamo la scelta più diretta, individuando le basi naturali dei corrispondenti spazi di catena e co-catena. Sotto l'identificazione selezionata, abbiamo che la matrice [δp.1], che rappresenta δp-1 nelle basi naturali di Cp-1 e Cp, è uguale alla trasposta della matrice [δp], che rappresenta δp nelle basi naturali di Cp e Cp-1 , in modo che [δp-1] = [δp ]t , e rappresentano l'identificazione e la dualità nel diagramma sottostante, dove gli spazi di co-catene sono rappresentati con indici inferiori: Cꓸ= (Cp, δp):= C3↔C2↔C1↔C0 dove δCp-1◦ δCp=0

Definizione 2.1.13 (Matrici operatori). Fissate le basi, cioè ordinati gli insiemi di -celle, le matrici di confine e gli operatori di confine, che vediamo nell'Esempio 2.1.1, sono determinati in modo univoco. Tali matrici sono molto sparse, con scarsità che cresce linearmente con il numero di celle (righe). La scarsità può essere definita come uno meno il rapporto tra diversi da zero e il numero di elementi di matrice. È giusto considerare che i non-zero per riga sono delimitati da una piccola costante nelle matrici topologiche, quindi il numero di elementi diversi da zero cresce linearmente con m. Con comuni strutture dati (Cimrman, 2015) per matrici sparse, il costo di archiviazione O(m) è lineare con il numero di celle, con O piccolo costo per cella che dipende dallo schema di archiviazione.

Esempio 2.1.1 (Matrice di confine). In questo piccolo esempio costruiamo la matrice sparsa [δ2] dalla moltiplicazione di matrici caratteristiche EV e FV trasposta (per i dettagli si veda DiCarlo et al. (2014)). Alcune matrici sono scritte trasposte qui per motivi di spazio. Mostriamo anche come sia facile estrarre sia il confine che eventuali cicli di un cellulare complesso, usando il semianello (Kepner et al., 2016) moltiplicazione matriciale . Attualmente, la moltiplicazione algebrica di matrici viene utilizzata nei nostri algoritmi sotto forma di una moltiplicazione di matrici standard (sparsa), seguita da un filtraggio di matrice risultante. È in corso il porting su Julia della libreria SuiteSparse:GraphBLAS di Tim Davis (2019) lo sviluppo dal nostro gruppo. I nomi di Julia composti da due caratteri da V, E, F, C (per 0, 1, 2, 3 celle) verranno usati spesso al posto dei simboli matematici δp e δq . In questo caso abbiamo sempre: YX : X → Y, e YX' = XY.

Definizione 2.1.14 (Basi di co-catena naturali). Negli algoritmi topologici, vedi Algoritmo 6, facciamo largo uso di mappe co-confine tra spazi di co-catene: δp ∶ Cp→ Cp+1. Mediante l'identificazione della co-catena e degli spazi della catena, vedere il diagramma (3), le mappe δp ∶ Cp→ Cp+1vanno tra le stesse basi naturali di δp ∶ Cp→ Cp-1, cioè da/a celle graduate.

2.2. Disposizione dello spazio La parola disposizione è usata nella geometria combinatoria e nella topologia computazionale come sinonimo di partizione spaziale. La costruzione di disposizioni di linee (Dimca, 2017), segmenti, piani e altri oggetti geometrici è discussa in Fogel et al. (2007), con una descrizione del software CGAL (Fabri et al., 2000), che implementa arrangiamenti 2D/3D con Nef polyhedra (Bieri, 1995) di Hachenberger et al. (2007). Una rassegna di documenti e algoritmi riguardanti la costruzione e il conteggio delle celle può essere trovata nel capitolo sugli Arrangiamenti nell'“Handbook of Discrete”. e geometria computazionale” (Goodman et al., 2017). Gli arrangiamenti di politopi, iperpiani e d-cerchi sono discussi in Björner e Ziegler (1992).

Definizione 2.2.1 (Disposizione spaziale). Data una collezione finita S di oggetti geometrici in Ed , la disposizione A(S) è la decomposizione di Ed in celle aperte connesse di dimensioni 0, 1,…,d indotte da S (Halperin e Sharir, 2017). Siamo interessati alla partizione spaziale euclidea indotta da un insieme di complessi cellulari PL.

Esempio 2.2.1 (Disposizione dello spazio). Le 2-celle irriducibili della partizione E2 generata da rettangoli casuali e da approssimazioni poligonali casuali del cerchio 2D. Prendere atto del fatto che 2 celle possono essere non convesse e/o non contrattile.

2.3. Geometria solida costruttiva Il termine Constructive Solid Geometry (CSG) è stato coniato da Voelcker e Requicha (1977) in uno dei rapporti fondamentali del Production Automation Project (PAP) presso la Rochester University. È usato per creare un oggetto complesso utilizzando operatori booleani per combinare alcuni oggetti primitivi. A volte è usato per descrivere proceduralmente la sequenza di operazioni eseguite da un designer durante le fasi iniziali di creazione della forma. Oggi è più usato per caratterizzare le strutture dati alla base dell'operazione UNDO all'interno di un guscio di progettazione interattiva, piuttosto che specificare la descrizione generativa di una forma complessa, ovvero come schema di rappresentazione (Requicha, 1980). Algoritmi di conversione della rappresentazione tra CSG e rappresentazioni di confine (B-reps) sono state fornite da Shapiro e Vossler (1993).

Definizione 2.3.1. La geometria solida costruttiva è uno schema di rappresentazione per modellare oggetti solidi rigidi come composizioni teoriche stabilite di "mattoni" solidi primitivi (Voelcker e Requicha, 1977). Nota. CSG può essere descritto come la combinazione di volumi occupati da oggetti 3D sovrapposti, utilizzando l'insieme di operazioni booleane. Tipiche primitive sono definite come una combinazione di semispazi delimitati da superfici quadriche generali (parallelepipedi, cilindri, sfere, coni, tori, cunei, solidi spazzati). Le operazioni sono unione, intersezione, differenza e complemento.

Nota. Altre due definizioni formali di CSG sono le seguenti:

(a) albero binario non completo con solidi primitivi in coordinate locali sulle foglie e mappe affini o operazioni booleane sui nodi non foglia;

(b) binario completo albero con booleane sulle non foglie e solidi primitivi in ​​coordinate universali (le coordinate della radice) sulle foglie.

In altre parole: il solido è rappresentato come unione, intersezione e differenza di solidi primitivi che sono posizionati in spazio dalle trasformazioni del corpo rigido (Hoffmann e Shapiro, 2017).

Algoritmo 2 (valutazione CSG a coppie). I metodi di valutazione CSG a coppie richiedono un attraversamento DFS post-ordine dell’albero delle espressioni e il calcolo sequenziale a coppie di ogni operazione booleana incontrata su ciascun nodo con il risultato parziale corrente, fino a quando non viene valutata l'operazione di root. Vedi ad esempio il calcolo n-ario dei booleani di Zhou et al. (2016). Altri tipi di attraversamento non modificano il numero e la complessità delle operazioni.

Nota. È ben noto che le operazioni booleane tra B-reps di due solidi A e B hanno complessità O(n2 ) nel caso peggiore, poiché ogni 2-faccia di A può intersecarsi con ogni faccia di B. Quindi è possibile dimostrare che la suddivisione è molto più efficiente di una gerarchica. Vedere Proprietà 2.3.1.

Nota. Anche i processi computazionali CSG n-ari standard (a coppie) mancano di robustezza poiché accumulerebbero errori numerici, che alla fine modificano la topologia del risultato parziale e fanno sì che le applicazioni si fermino facilmente per errore. Quindi, le rappresentazioni di confine intermedie devono essere attentamente curate, prima di continuare l'attraversamento.

Algoritmo 3 (valutazione flat CSG). In questo articolo daremo una nuova organizzazione del processo computazionale per valutare le espressioni CSG di qualsiasi profondità e complessità. La valutazione di un'espressione CSG di qualche complessità è fatta qui con un diverso approccio computazionale. L'albero stesso viene utilizzato solo per applicare trasformazioni affini a solidi primitivi, al fine di ridimensionarli, ruotarli e tradurli nelle loro posizioni e atteggiamenti finali ("mondo"). Tutte le loro 2-celle sono così accumulate in un'unica raccolta, e ciascuna di esse è efficacemente frammentata contro le altre, generando così una collezione di topologie locali indipendenti che vengono fuse per congruenza di confine, usando una singola identificazione con arrotondamento di vertici vicini. La partizione dello spazio globale viene quindi generata e tutti i solidi iniziali sono classificati rispetto a tali 3 celle, con un unico test di contenimento point-set. Infine, qualsiasi forma booleana di complessità arbitraria, utilizzando gli stessi componenti solidi, può essere immediatamente valutata mediante operazioni logiche vettorizzate bit per bit.

Proprietà 2.3.1 (Complessità della valutazione). La complessità del caso peggiore dell'approccio piatto è migliore di quella gerarchica uno, come si vede in un grafico estremamente semplificato (vedi Osservazione dopo l'algoritmo 2), supponendo che su ogni coppia di facce un il taglio singolo viene eseguito in tempo costante, Nel caso di 푚 oggetti solidi, ciascuno con 푛 confine 2 celle, nell'ipotesi piatta che ogni 2-cella è intersecata da tutte le altre 푚푛− 1, abbiamo un numero polinomiale 푂(푚2푛 2 ) di eventuali tagli. Al contrario, nel caso gerarchico, accade che ogni 2 celle intersechi tutte quelle precedentemente divise, richiedendo un esponenziale numero 푂(2푚푛) di tagli. Anche per = 2 e 푛 = 6, è 푚2푛 2 = 144 e 2 푚푛 = 4096. Naturalmente, questo estremo L'ipotesi che ogni faccia tagli tutte le altre, non troverà mai applicazione nella pratica, ma il confronto sembra comunque utile.

Algoritmo 4 (CSG mediante ray-casting). Un tipico algoritmo approssimativo per la valutazione di un albero CSG per la visualizzazione grafica è per mezzo di ray-casting, e interseca per ogni raggio le superfici limite appropriate, da calcolare per insieme di appartenenza funzione di classificazione (SMC). Questo algoritmo discreto richiede una corretta combinazione booleana della suddivisione del raggio in segmenti di linea (Requicha e Tilove, 1978).

Esempio 2.3.1 (esempio CSG — 1/2). Di seguito è riportato un esempio di geometria solida costruttiva (da Wikipedia). La formula booleana è costruita utilizzando tre solidi primitivi (cilindro, cubo, sfera). L'albero delle espressioni è mostrato con operatori sulle non foglie e solidi primitivi sulle foglie. Nel nostro approccio l'assieme Struct, con parametrizzato funzioni generatrici e trasformazioni affini (scala, rotazione, traslazione) viene utilizzato per il posizionamento 3D del 5 termini X1 ,X2 ,X3 ,Y,Z, risp.

2.4. Algebra booleana In matematica e logica matematica, l'algebra booleana è il tipo di algebra in cui i valori delle variabili sono i valori di verità vero e falso, solitamente indicati rispettivamente con 1 e 0. In questo documento rappresentiamo e implementiamo per d = 2, 3 le algebre booleane solide di CSG lineari a tratti con celle regolari chiuse, generate dalla disposizione di Ed indotto da un insieme di complessi cellulari con cellule poliedriche di dimensione d− 1 (Paoluzzi et al., 2017b). In particolare, mostriamo che la struttura di ogni termine di questa algebra è caratterizzata da un insieme discreto di punti, ciascuno calcolato una volta per tutte all'interno di ogni atomo. Classificazioni di appartenenza all’insieme (SMC) rispetto a tali singoli punti interni degli atomi, calcola la struttura di ogni elemento di algebra, e in particolare trasforma ogni input di ogni formula booleana, in un array logico di lunghezza m , pari al numero di atomi.

Definizione 2.4.1 (Algebra finita). Sia A un insieme non vuoto e le operazioni ⊗i ∶ Ani → A funzioni di argomento ni . Il sistema <A; ⊗1,.., ⊗k> si chiama algebra. In alternativa diciamo che A è un insieme con operazioni ⊗1,.., ⊗k . Per definizione, un'algebra A è chiusa rispetto alle sue operazioni. Se A ha un numero finito di elementi si dice algebra finita.

Definizione 2.4.2 (Generatori). Un insieme H genera l'algebra A (in alcune operazioni) se A è il più piccolo insieme chiuso rispetto alle operazioni e contenente H. Gli elementi ℎi∈ H sono detti generatori o atomi dell'algebra A.

Definizione 2.4.3 (Algebra booleana). Possiamo pensare alle algebre booleane ℬ come un insieme che è isomorfo all'insieme delle potenze P(X) di un insieme finito X. L’insieme delle potenze è naturalmente dotato di operazioni di complemento, unione e intersezione, che corrisponde alle operazioni −, ∨, ∧ dell'algebra booleana.

Proprietà 2.4.1 (Algebra booleana). Qualsiasi algebra ℬ con n generatori è isomorfa all'algebra booleana con insieme P(X) , contenente n elementi. Pertanto ℬ può essere mappato uno a uno con l'insieme χX(P(X) )delle immagini di funzioni caratteristiche di ℬ elementi rispetto a X, cioè con l'insieme di stringhe di n elementi in {0, 1}.

Definizione 2.4.4 (Atomo). Un atomo è qualcosa che non può essere scomposto in due sottoinsiemi propri, come un singleton che non può essere scritta come unione di due sottoinsiemi strettamente più piccoli. Un atomo è un elemento minimo diverso da zero; a è un atomo se e solo se per ogni b, o b ∧ a = a oppure b ∧ a = 0. Nel primo caso diciamo che a appartiene alla struttura di b.

Definizione 2.4.5 (Struttura degli elementi di algebra). Chiamiamo struttura di b ∈ P(X) il sottoinsieme di atomi S tale che b sia unione irriducibile di S . Per estensione, chiamiamo anche struttura di b la stringa binaria associata alla sequenza ordinata dei suoi atomi (elementi di ). In altre parole, la struttura S(b) è l'immagine della funzione caratteristica χX (S) .

Esempio 2.4.1. Sia P un insieme di m iperpiani che partizionano Ed in celle convesse relativamente aperte di dimensione che vanno da zero (punti) a d . La raccolta di tutte queste celle è chiamata disposizione lineare A(P) ed è stata studiata estesamente in geometria computazionale (Edelsbrunner, 1987). Le celle in A(P) sono atomi di un'algebra booleana di sottoinsiemi di Ed  che può essere formato dall'unione di celle convesse nella disposizione.

Esempio 2.4.2 (disposizione dello spazio 2D). generato in E2da due 2-complessi sovrapposti. La figura (c) sotto mostra una partizione di E2in quattro sottoinsiemi irriducibili: la regione rossa A, la regione verde B, la regione sovrapposta e la regione esterna A ∪ B, cioè il resto del piano. L'insieme degli atomi dell'algebra booleana ℬ è lo stesso: le regioni colorate sono i quattro atomi dell’ algebra ℬ, e ci sono 24 = 16 elementi distinti S∈ ℬ. La struttura di ogni elemento S∈ ℬ è un'unione di ℬ atomi; come una catena in C2 , è una somma di elementi di base.

Esempio 2.4.3 (Algebra booleana). Consideriamo le colonne della Tabella 1. Per colonne, la tabella contiene la coordinata rappresentazione delle 2-catene A,B e Ω = A ∪ B nello spazio delle catene C2 . Nel linguaggio della topologia algebrica:

A=c2+c3, B= c3+c4, Ω= c1 = -( c2+c3+c4).

L'algebra booleana degli insiemi è qui rappresentata dall'insieme delle potenze A = P (U2), con U2 = { c1, c2, c3, c4}, con 24 = 16 termini binari di lunghezza # U2= 4 sono dati nella Tabella 2, insieme alla loro interpretazione semantica in algebra degli insiemi. Naturalmente, il vettore di coordinate che rappresenta l'insieme universale, cioè l'intero spazio topologico X ∶= E2 generato dalle colonne [δ2] è data da [X] = (1, 1, 1, 1), inclusa la cella esterna. Ricordiamo che il complemento di A, indicato con −A , è definito come X⧵A e che l'operazione differenza A⧵B è definita come A∩ -B.

Esempio 2.4.4 (Algebra solida 3D). Ogni 3-cella è generata da una colonna della matrice sparsa della mappa di confine δ3 ∶ C3 → C2 , con valori in {−1, 0, 1}. Colonne di [δ3 ] sono 2 cicli, cioè catene chiuse in C2 . Infatti, le catene chiuse non hanno confine. Le 3-celle sono gli atomi irriducibili dell'algebra CSG con celle regolari chiuse. Possono essere non contrattili in un punto (quando contengono un foro) e non convessi. La cellula esterna è il complemento della loro unione. Qualsiasi modello dell'algebra booleana (su 2 25 – vedi figura sotto) del CSG generato da questi cinque cubi è costituito dagli stessi 25 atomi.

2.5. Algebre Topologiche Shapiro (1991) ha presentato una gerarchia di algebre per definire formalmente una famiglia di rappresentazioni teoriche degli insiemi finiti (FSR) di sottoinsiemi semi-algebrici di Ed , compresi molti schemi di rappresentazione noti per oggetti solidi e non solidi, come rappresentazioni di confine, geometria solida costruttiva, decomposizioni cellulari, geometria selettiva di complessi e altri. Le applicazioni esemplificative includevano conversioni B-rep → CSG e CSG → B-rep. In particolare, una gerarchia di algebre su sottoinsiemi semi-algebrici di Ed è discusso, dove gli elementi di algebra possono essere costruiti usando un numero finito di operazioni FSR (ops set standard, chiusura e componente connessa).

Definizione 2.5.1 (Catena come base topologica). Una base ℬ per uno spazio topologico X con topologia T è un sottoinsieme di insiemi aperti in T tali che tutti gli altri insiemi aperti possono essere scritti come unione di elementi di ℬ. Un insieme si dice chiuso se il suo complementare è aperto. Quando sia un insieme che il suo complemento sono aperti, vengono chiamati clopen. Consideriamo entrambe le d-celle e catene come insiemi di punti. Le catene in uno spazio lineare Cd possono essere visti come insiemi di punti aperti generati da somma di spazi topologici.

Proprietà 2.5.1 (Spazio della catena come topologia discreta). Si può dimostrare che, se tutte le componenti connesse di uno spazio sono aperti, allora un insieme è clopen se e solo se è l'unione di componenti connessi disgiunti a coppie. Tutte le catene in C= ⊕d Cd sono clopen. Ancora di più, tutti gli insiemi Cp di tutte le p-catene , incluso Cp stesso e l’insieme vuoto, hanno la topologia discreta , poiché tutti gli elementi dell'insieme di potenze X= P (Ud ) sono clopen, dove Ud è la base.

Proprietà 2.5.2 (Spazi di catena come algebre booleane). L'insieme X di sottoinsiemi di d-catene indipendenti può rappresentare uno spazio topologico discreto. Usando l'unione e l'intersezione come operazioni, i sottoinsiemi clopen di uno spazio topologico discreto X formano un'algebra booleana finita. Si può dimostrare che ogni algebra booleana finita è isomorfa all’algebra booleana ℬ di tutti i sottoinsiemi di un insieme finito (Halmos, 1963), e può essere rappresentata dal 2 N termini binari in ℬ . Nel nostro caso N = Dim Cd= #Ud .

3. Pipeline computazionale Ogni forma booleana dell'algebra PL solida studiata in questo articolo è valutata dal seguente calcolo : (a) calcolo della disposizione dello spazio Ed indotto dai poliedri di input, seguito da (b) il calcolo di un array sparso di bit, che denota la struttura atomica, per ogni variabile solida di input, e da (c) l'applicazione nativa di operatori bit a bit su stringhe di bit, secondo la formula simbolica da risolvere. Una sola decomposizione di Ed può essere usata per valutare ogni forma funzionale sulla stessa algebra, inclusa ogni sottoespressione tra gli stessi argomenti (vedi Esempio 3.6.5).

3.1. Suddivisione di 2 celle Ogni 2 celle σ nell’insieme di input S2 si interseca indipendentemente con le altre di possibile intersezione, producendo il suo arrangiamento dello spazio E2 come una catena 2-complesso C**ꓸ** (σ). Le 2 celle di possibile intersezione con σ sono quelle le cui scatole di contenimento intersecano la scatola di σ.

Algoritmo 5 (Splitting di 2 celle in 2D). Ogni insieme I(σ) di 2 celle di possibile intersezione con σ, è calcolato in modo efficiente per intersezione combinatoria dei risultati dell'interrogazione su d diversi (uno per ogni coordinata) alberi di intervallo (Preparata e Shamos, 1985). Ogni insieme I(σ) , per σ∈ S2 , è mappato in modo fine in E3, portando σ al sottospazio z = 0. l'arrangiamento A(σ)=C**ꓸ** (σ). viene prima calcolato in E2, e quindi mappato di nuovo in E3 .

Nota. L'intersezione effettiva di 2 celle viene calcolata tra i segmenti di linea in 2D, mantenendo solo la parte non esterna del sottografo massimo 2-connesso dei risultati, ottenendo così una decomposizione regolare 2-complessa di 2-celle . Vedere Figura nell'esempio 3.1.1. Questo calcolo viene eseguito indipendentemente per ogni di 2-cella σ nei complessi cellulari di input. Per accelerare fortemente il calcolo, la scissione può essere calcolata tramite i canali di Julia, che sono in grado di prendere vantaggio di entrambi i nodi di calcolo locali e remoti, facendo uso di questo approccio basato sui dati in modo parallelo.

Proprietà 3.1.1 (Complessità temporale). La complessità dell'algoritmo 5 di suddivisione a 2 celle può essere considerata nel caso medio lineare nel numero totale di 2 celle, moltiplicato per un fattore log n, necessario per calcolare quelle di probabile intersezione, quando la scissione di una singola cella può considerarsi fatta in tempo costante. Infatti, normalmente, ogni cella è piccola con rispetto alla scena, ed è intersecata da un piccolo numero di altre celle. Sotto questa ipotesi sia la mappatura da/a z = 0 e ogni intersezione di 1 cella viene eseguita in tempo costante e in numero ridotto. Inoltre, la creazione 2D di celle di complessi produce due piccole matrici sparse δ0(σ) e δ1(σ) per 2 celle σ . I numeri delle loro righe e le colonne sono quelle dei vertici, dei bordi e delle facce della divisione σ, cioè numeri limitati molto piccoli. La complessità dell'algoritmo 5 è quindi O(lk (n log n)), con l numero medio di celle componenti prodotte (a seconda di entrambe le dimensioni e numero di incidenza di ogni σ) e 0 ≤ k ≤ 1, fattore di distribuzione inversamente proporzionale al numero di nuclei utilizzati. La costruzione iniziale degli alberi di intervallo ha la stessa complessità del (doppio) ordinamento degli intervalli.

3.2. Confezione regalo topologica in 2D Per calcolare le 2 celle come 1 ciclo e le corrispondenti matrici [δ2], l’algoritmo “ Confezione Regalo Topologica” (TGW) (Paoluzzi et al., 2017b) viene applicato in 2D alla matrice [δ1] (σ), per ogni σ. Una costruzione passo passo di un 1-ciclo a partire da una singola 1-cella all'interno di un 1-complesso, è mostrato nell'Esempio 3.2.1. Una descrizione completa di questo algoritmo e il relativo pseudocodice sono riportati in Paoluzzi et al. (2017a) e Paoluzzi et al. (2017b).

Osservazione (Significato delle colonne della matrice). Ricordiamo innanzitutto che la matrice di una trasformazione lineare tra due spazi lineari è determinato in modo univoco una volta fissate le basi del dominio e gli spazi di destinazione. Quindi nota che le colonne della matrice sono la rappresentazione coordinata dell'elemento base del dominio, rappresentata come combinazione lineare della base del spazio di destinazione. In particolare, le colonne contengono i coefficienti scalari di tali combinazioni lineari, nel nostro caso presi da {−1, 0,+1}. Quindi, una d-cella di base è rappresentata dal suo confine, una combinazione lineare di celle di base (d-1).

Algoritmo 6 (Confezione regalo topologica). Riassumiamo qui l'algoritmo TGW in un d-spazio. L'input è la matrice sparsa[d-1]; l'output è la matrice [d+] ∶ Cd → Zd-1, da d-catene a (d-1)-cicli (vedi Sezione 3.5).

1. Initialization: m,n = [d-1].shape; marks=zeros(n) ; [d+] = [] ;

2. while sum(marks) < 2n do

(a) select the (d − 1)-cell seed of the column extraction

(b) compute boundary cd-2 of seed cell

(c) until boundary cd-2 becomes empty do

i. corolla = []

ii. for each “stem” cell τ∈ cd-2 do

a. compute the τ coboundary

b. compute the new petal cell

c. orient petal and insert it in corolla

iii. insert corolla cells in current cd-1

iv. compute again the boundary of cd-1

(d) update the mark counters of used cells

(e) append a new column to [d+]

3.3. Complesso della catena di congruenza Introduciamo in questa sezione un modo semplice per incollare insieme un insieme di informazioni geometriche e topologiche locali generato indipendentemente da ogni ingresso 2-cella, dando poche definizioni di vicinanza e congruenza, che consentirà di trasformare l'unione di topologie locali in un'unica “topologia quoziente” globale. Per una discussione più approfondita il si rimanda a DelMonte et al. (2020).

Definizione 3.3.1 (-vicinanza). Diciamo che due punti u, v ∈ V sono -vicini e scriviamo u ∼ v, quando il la loro distanza euclidea è d(u, v) ≤ 2. Sia Si il sottoinsieme dei punti a distanza minore di da vi . La -vicinanza ∼ è una relazione di equivalenza, poiché è riflessiva, simmetrica e transitiva. In particolare, è transitiva poiché ogni coppia di punti u, v ∈ Si sono -vicini, perché entrambi hanno una distanza minore di da vi , e quindi hanno una distanza non superiore a 2 da ciascuno Altro. Più formalmente, se u ∼ v e v ∼ w, allora u ∼ w, poiché la distanza da vi di ogni punto (es., u,v, w) in Si è meno di .

Definizione 3.3.2 ( -congruenza). Diciamo che due p -celle e, f sono -congruenti, e scriviamo e≅ f, quando esiste un biiezione μ tra le loro 0-facce che mappano a coppie i vertici in –vertici vicini.

Definizione 3.3.3 (Topologia quoziente). Lo spazio quoziente di uno spazio topologico X sotto una data relazione di equivalenza è un nuovo spazio topologico costruito dotando l'insieme quoziente di X della topologia quoziente. Questa è la topologia più fine che rende continua la proiezione canonica ∶ x ↦ [x], cioè la funzione che mappa i punti nella loro classe di equivalenza.

Nota (Topologia quoziente). Il quoziente o la topologia di identificazione fornisce un metodo per ottenere una topologia su X / da una topologia su X. La topologia quoziente (T ) è esattamente quella che fa "assomigliare" lo spazio risultante all'originale , con gli elementi identificati incollati tra loro.

Proprietà 3.3.1 (Congruenza catena complessa). La -congruenza tra le catene elementari (dette cellule) c,d ∈ Cp, indicata con c≅ d, è una relazione di equivalenza graduata, così che un complesso di catene (Cp , δp) può essere rappresentato da uno molto più piccolo , che chiamiamo complesso di catene di congruenza, con (Cp , δp ) = (Gp , δp’ ), dove Gp = (Cp ) =Cp/ , e dove δp’ = δp◦ ∶= (Cp+1 )= δp : Gp → (Gp+1)

Dati 3.3.1 (Matrice accumulatore diagonale a blocchi). Durante il calcolo della disposizione dello spazio generato da una collezione di complessi cellulari, siamo partiti dal calcolo indipendente delle intersezioni di ogni singolo ingresso 2-cella con gli altri. La topologia di queste intersezioni è codificata all'interno di un insieme di complessi di catena (0-2)-dimensionali, memorizzati all'interno di matrici di accumulatori [Δ1 ] ∶ C1→ C2 e [Δ0 ] ∶ C0 → C1 . Entrambe le matrici di accumulatori [Δ1 ] ∶ C1→ C2 e [Δ0 ] ∶ C0 → C1 hanno una struttura diagonale a blocchi sparsi, con due livelli nidificati di blocchi diagonali (sparsi). Ogni blocco esterno riguarda uno degli m oggetti geometrici di input, e nk blocchi interni, 1 ≤ k ≤ m. Chiamiamo [h], 1 ≤ ℎ ≤ m, i blocchi esterni (grigio chiaro), e [k], 1 ≤ k ≤ mh , i blocchi interni (grigio scuro), dove mh è il numero di 2 celle nel k-esimo oggetto geometrico di input. Nella specifica dell'algoritmo di seguito, vengono utilizzati rispettivamente un array denso W e due array sparsi Delta\_0, Delta\_1 per le coordinate dei vertici di input e i repository di matrici sparse a blocchi [Δ0 ] e [Δ1 ].

Algoritmo 7 (Chain Complex Congruence (CCC)). Abbiamo discusso sopra il marshalling diagonale del blocco [Δ0 ] e [Δ1 ] di matrici di co-confine locali. L'obiettivo dell'algoritmo CCC è di unire le catene locali usando le relazioni di equivalenza di -congruenza tra 0, 1 e 2 celle (catene elementari). In particolare, riduciamo le matrici di co-confine blocco-diagonale [Δ0 ] e [Δ1 ], utilizzati come accumulatori di matrici di catene locali di co-confine, alle matrici globali[δ0] e [δ1], rappresentative della topologia di congruenza, cioè della congruenza quoziente tra tutte le 0-,1-,2-celle, tramite operazioni algebriche elementari sulle loro colonne.

1. Scopriamo la -vicinanza dei vertici chiamando la funzione CSG.vcongruence(V::Matrix;epsilon=1e-6) sull'ingresso 3 ×n delle coordinate 3D, che restituisce i nuovi vertici W e la mappa delle vclassi di -congruenza. La matrice W 3 × m contiene le coordinate dei rappresentanti di classe, mappate su ciascun centroide di classe.
2. Con la funzione Julia CSG.cellcongruence sostituiamo ogni sottoinsieme di colonne della matrice sparsa Delta\_0 corrispondenti ai vertici -vicini, con il loro baricentro. Una nuova matrice viene prodotta dall'array di nuovi vettori. Infine, righe uguali di questa nuova matrice, scoperta tramite un dizionario, sono sostituite da un unico rappresentante.
3. La stessa funzione CSG.cellcongruence è applicata anche a [Δ1 ], sommando ogni sottoinsieme di colonne corrispondente a ciascuna classe di archi congruenti, generando così una nuova matrice sparsa di Julia dall'insieme risultante di colonne. Quindi, riduciamo ogni sottoinsieme di righe uguali, se presenti, a una singola riga rappresentativa di facce congruenti.
4. Infine, una funzione Julia di livello superiore CSG.chaincongruence mappa i dati di input W, Delta\_0, Delta\_1, in un rappresentazione compatta V, EV, FE del complesso a catena V ∶ C0 → E3 , δ0 ∶ C0→ C1 e δ1 ∶ C1 → C2.

Proprietà 3.3.2 (Robustezza topologica). Vorremmo sottolineare che le 1-celle scomposte (cioè i bordi di input) e i rappresentanti delle loro classi di congruenza (cioè archi di output) sono associati uno a uno con le righe di [Δ0 ] → [δ0] e le colonne di [Δ1 ] → [δ1], rispettivamente, soddisfacendo così i vincoli topologici [Δ1 ][Δ0 ] = [δ1][ δ0 ] = [0], che sono controllati come invarianti nei nostri test (vedi Algoritmo 8).

Proprietà 3.3.3 (Complessità temporale). L'algoritmo che costruisce un kd-albero bilanciato per eseguire la ricerca per intervallo ha una complessità nel caso peggiore di O(kn log n). Il passaggio 1. dell'algoritmo 7 richiede una ricerca a intervallo singolo di un kd-albero , con dimensione dell’intervallo , eseguito in un singolo attraversamento di alberi, quindi in O(n), con costruzione ad albero in O(dn log n). I calcoli del centroide locale vengono eseguiti in tempo costante atteso per produrre m vertici, per un tempo totale O(n + m). Ciascuna delle d− 1 iterazioni della funzione CSG.cellcongruence, che esegue la trasformazione [Δp ] → [δp], deve riscrivere una matrice più piccola da una più grande, con coefficienti di scala per quantità diverse da zero, sia su righe che su colonne, che vanno da 3 a 10 in media. Sembra corretto stimare il tempo medio totale per O(kd − tree) + O(centroid) + O(out\_writing) = O(dn log n + m + nnz), dove nnz (non zero) è uguale a k1m (3 ≤ k1 ≤ 10), cioè proporzionale ai vertici finali.

Algoritmo 8 (Invarianti topologici). Gli invarianti sono predicati (funzioni che restituiscono un valore booleano) che devono essere soddisfatti dai valori correnti delle variabili in punti specifici durante il processo di valutazione del programma. In particolare, gli invarianti topologici vengono valutati dinamicamente durante l'esecuzione per rilevare errori numerici comuni. L'algoritmo TGW è particolarmente fragile rispetto a errori topologici di questo tipo, per cui vengono valutati pochi invarianti in vari punti della condotta. L'invariante più comune è la caratteristica topologica, o numero di Eulero, entrambi in 2D e in 3D. Infatti abbiamo V − E + F = 2 e V − E + F − C = 0 sulla 2-sfera e sulla 3-sfera, topologicamente equivalente rispettivamente al piano e allo spazio. Si calcolano quindi alcuni invarianti:

1. Prima della scissione 2D, per ogni ingresso 2-faccia, abbiamo una insieme di segmenti di linea da intersecare tra loro, con E ≤ 2V;

2. Dopo la suddivisione 2D e TGW, per ogni componente a 2 facce semplicemente connesso, è necessariamente V − E + F = 2.

3. Prima della congruenza complessa della catena, quando si costruiscono le matrici contenitore sparse [Δ0 ] e [Δ1 ] , [Δ1 ][Δ0 ] = 0

4. Dopo CCC dobbiamo avere, per la topologia quoziente, che [δ1][ δ0] = 0 vale, con l= V colonne di [δ0 ];

5. Prima di TGW in 3D, con ingresso [2 ]m,n = [δ1 ]T abbiamo m= E, n = F, e l − m + n = p;

6. Dopo TGW in 3D, deve valere l'identità V − E + F − C = 0, dove C = p.

3.4. Confezione regalo topologica in 3D L'input in E3 è la matrice sparsa [2 ]; l'output è la matrice sparsa [3+ ] dell'operatore3+: Z3 → Z2 a partire dai 3-cicli irriducibili a 2-cicli (vedi Algoritmo 9).

Proprietà 3.4.1 (Complessità della confezione regalo topologica). La complessità temporale dell'algoritmo TGW è quella necessaria per scrivere la matrice del ciclo [3+], cioè calcolare i suoi termini nnz (diversi da zero). Cercando uno pseudocodice dettagliato dato in (Paoluzzi et al., 2017b), si può vedere che: se n è il numero di d-celle e m è il numero di (d − 1)-celle, la complessità temporale di questo algoritmo è O(nm log m) nel caso peggiore di complessità illimitata di d-celle, e grosso modo O(nk log k) se la complessità dei (d − 1)-cicli è limitata da una costante k.

Osservazione ( d− 2-celle “staminali”). Osserviamo che il Le cellule “staminali” sono tutte le d− 2-celle unità nel parziale c , con d∈ Cd-1. Naturalmente, tali cellule staminali sono sempre due quando d = 2, e almeno tre quando d= 3.

3.5. Cicli e confini

Definizione 3.5.1. Un p-ciclo è definito come una p-catena senza confine, quindi è un elemento del kernel Zp di p.

Proprietà 3.5.1 (Le colonne della matrice 3sono a 2 cicli). Come abbiamo visto nella sezione precedente, l'algoritmo TGW in 3D produce la matrice sparsa [3], partendo dalla matrice sparsa [2] del 2-scheletro della partizione di E3 indotta dai dati di input, cioè da una raccolta Sd-1 di complessi cellulari. Ogni colonna della matrice [3] , diciamo la k-esima colonna [uk ], è un 2-ciclo per costruzione, poiché la sua costruzione come 2-catena si ferma quando [2][ uk] = [0] , cioè quando il suo confine è vuoto.

Proprietà 3.5.2 (Righe della matrice 3 somma a zero). Ogni riga della matrice 3corrisponde a un elemento irriducibile nella base U2 ⊂ C2. Per costruzione nell'algoritmo TGW, ogni elemento di base u ∈ U2 è usato esattamente due volte nella costruzione del confine di 3-celle, con coefficienti opposti +1 e -1, dimostrando così l'asserzione. In altre parole, l'insieme Z2 di 2 cicli prodotti da TGW in 3D non è linearmente indipendente. In particolare, ognuno di essi è generato dalla somma topologica degli altri. Nota (i cicli non si intersecano). Due osservazioni sono molto importanti per l'algoritmo 9, riguardo alle trasformazioni di cicli di catene alle catene di confine. Il primo è che, per costruzione, i 2-cicli corrispondenti alle colonne di [3]sono (a) elementare (irriducibile) e (b) non intersecante; la seconda riguarda il loro numero. È ben noto che Zp⊇ Bp o, a parole, possono esserci cicli che non sono confini. Esempio 3.5.1 (Contorno di sfere concentriche). Considera la partizione spaziale 3D generata da due 2-sfere S1 e S2 con lo stesso centro e raggi diversi r1 > r2. Ci sono tre celle solide: (a) la cella esterna, cioè E3 meno la sfera di raggio r1 ; (b) la sfera intermedia solida con foro sferico all'interno e spessore r1 – r2 ; e (c) il solido sfera interna di raggio r2. Ci sono quattro 2-catene (cicli) irriducibili chiuse generate come colonne di [3] da TGW in questo complesso, sommando a coppie allo zero e con orientamenti opposti. Indichiamo ordinato, dall'esterno all'interno, come [3] = [u1 u2 u3 u4]. Se indichiamo i tre elementi base “solidi” in U3⊂ C3 (3-catene) come A,B,C e l'intero spazio come X, possiamo esprimili come espressione di algebra booleana:

A= X − B − C; B = X − A – C; C = X − A – B.

In termini di 2-catene orientate è facile vedere che A=u1; B=u2+u3; C=u4, con u1+ u2 = u3 + u4 =0 e u1+u2+u3 + u4 =0.

Infine, notiamo che i cicli u2e u3 non sono confine di nessuna 3-catena . Nella scrittura più formale, abbiamo: u2, u3 ∈ (Z2 – B2 ) ⊂ C2 .

Esempio 3.5.2 (Cicli → confini). L'assieme Lar.Struct([ tube, L.r(pi/2,0,0), tube, L.r(0,pi/2,0), tubo ]) di tre istanze di cylinder() con n = 16 lati, raggio predefinito di lunghezza 1, altezza ℎ = 2 e k = 2 di seguito sono riportate le scomposizioni in direzione assiale. L'unità esplosa 2-catene della disposizione spaziale, e gli atomi corrispondenti alle 20 colonne di [3+] e le 19 di [3] sono mostrati. Al centro il confine esterno. Considera il numero di facce da dividere (288 vs 54) con due diverse strutture di dati: triangolazioni di confine e LAR.

Nota (shell non intersecanti). Nella modellazione solida, la parola shell è usata per denotare le superfici di contorno connesse di un oggetto solido (Paoluzzi et al., 1989). Qui un guscio diventa uno dei 2 cicli del confine di un massimo componente connesso di una 3-catena, o semplicemente: uno dei cicli al contorno di un massimo 3-componente. L'intero insieme di gusci, compresi quelli interni a qualche unità 3-catena (atomo), si ottiene come 2-catena per moltiplicazione matriciale di [3] volte, cioè le volte che la rappresentazione coordinata del solido (intero) come una 3-catena.

Osservazione (Test di correttezza). Il numero N di colonne della matrice [3] fornisce la dimensione dello spazio confine E3 , cioè il numero di elementi irriducibili dell'insieme 3. Si ottiene il 2-ciclo di confine della cella esterna per moltiplicazione di [3] volte un vettore colonna [**1**] con unità, pari alla somma dei 2 cicli esterni dei componenti complessi. Certo, è [2][3][ **1**] = [**0**], dove il numero di zeri equivale al numero di 1- celle del complesso.

Algoritmo 9 (Dai cicli ai confini). Qui discutiamo come ridurre la matrice [3] di 2 cicli irriducibili, alla matrice [3] di confini di 3-catene irriducibili (atomi) di E3 indotto dall'input. Per costruzione, ogni 3-catena unità irriducibile u ∈ U3 ⊂ C3 corrisponde a una singola cella connessa PL 3, possibilmente non convessa e non necessariamente contraibile per via di un punto. In altre parole, il confine di una 3-catena potrebbe eventualmente essere non connesso, e realizzato da uno o più 2-cicli.

1. Cercare, per ogni componente connesso del 2-complesso generato dalla congruenza, la colonna esterna [3];

(a) Ripetere la ricerca e l'eliminazione per ogni matrice [3+]k di una componente.

i. In ogni [3+]k cerca il ciclo che contiene il maggior numero di non zero (cioè il più alto numero di 2 celle), ii. Prima dell'eliminazione, verificare che il sottoinsieme di vertici coinvolti contenga i valori estremi (max e min) per ogni coordinata.

iii. Se vero, rimuovere questo ciclo dalla matrice del componente. Nell'improbabile caso opposto, il secondo il ciclo più lungo sarà candidato, e così via.

2. Tutte le matrici dei componenti [3]k senza il ciclo esterno locale (1 ≤ k≤ n), sono composti insieme in una matrice a blocchi sparsi [3] = [[3]1 ⋯[3]k ⋯ [3]n].

3.6. Algebra dei solidi Il secondo passo del nostro approccio alla geometria solida costruttiva consiste nel generare una rappresentazione degli argomenti solidi come combinazioni lineari di 3-catene indipendenti, cioè di 3-celle della partizione spaziale generata dall'input.

La base U3 delle 3-catene è rappresentata dalle colonne della matrice 3 . Questo insieme di 2 cicli può essere visto come una collezione di insiemi di punti, compreso l'intero spazio e l'insieme vuoto. In questo senso genera sia una topologia discreta di E3 e, tramite il teorema di rappresentazione di Stone(Stone, 1936), un'algebra booleana finita ℬ su C3 elementi.

Nota. Questa seconda fase del processo di valutazione di una forma funzionale comprendente modelli solidi e operatori di insieme è molto più semplice della prima fase, che consiste nell'eseguire una serie di test di classificazione dell'appartenenza al set (SMC), calcolabile in parallelo, per costruire una rappresentazione dei termini di input nell'algebra degli insiemi , usando array sparsi di bit.

3.6.1. Algebra degli insiemi Qualsiasi insieme finito di insiemi chiusi rispetto alle operazioni di teoria degli insiemi forma un'algebra booleana, cioè un reticolo distributivo, gli operatori join e meet sono unione e intersezione e l'operatore complemento è impostato complemento. L'elemento inferiore è e l'elemento superiore è l'insieme universale in esame. Da Stone(1936), ogni algebra booleana di N atomi - e quindi l'algebra booleana di oggetti solidi chiusi sotto unione regolarizzata e intersezione - è isomorfa all'algebra booleana degli insiemi su {0, 1}N . Qui, cerchiamo la rappresentazione binaria di oggetti solidi in termini di una formula geometrica solida, cioè per i coefficienti dei loro componenti in base Ud ⊂Cd .

Nota (rappresentazione di Julia). Se tutte le catene in Ud sono equiorientate, allora tali coefficienti sono semplicemente ricavati da {0, 1}, e ogni vettore di coordinate per Cd elementi è una sequenza binaria in {0, 1}N , con N = # Ud . In Julia implementiamo questa rappresentazione di termini algebrici utilizzando array di tipo BitArray, o array sparsi di tipo Int8, consumando pochi byte per elemento diverso da zero.

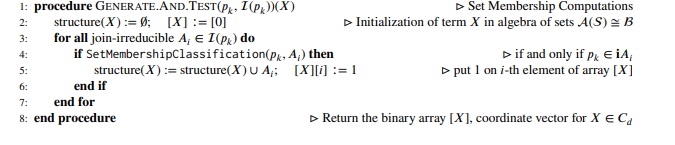
3.6.2. Atomi booleani Proprietà

3.6.1 (Gli atomi booleani sono unità 3-catene). Esiste una trasformazione naturale tra le d-catene definite su una disposizione spaziale e l'algebra generata da tale disposizione. Le d-catene di unità corrispondono agli atomi dell'algebra; la rappresentazione delle coordinate [c] (array di bit) di qualsiasi d-catena c genera la rappresentazione delle coordinate nello spazio limite [d][c] = [b] ∈ Bd-1 ⊂ Cd-1.

Congettura 3.1 (CSG dimensionale superiore). Il principale risultato introdotto da questo lavoro è la rappresentazione degli atomi dell'algebra booleana CSG generata da una partizione dello spazio E3, con la base U2 dello spazio C3rappresentato dalle colonne della matrice [3] come possibilmente 2-cicli non connessi di facce (2-celle)di E3. Lo stesso vale, ovviamente con indici scalati, per l'algebra CSG bidimensionale, e probabilmente dovrebbe valere in dimensioni superiori.

3.6.3. Algoritmo di generazione e test Viene utilizzata la rappresentazione di elementi irriducibili da join di L(S) ≅ A(Sd-1) come insiemi di punti discreti (si veda la Sezione 3.6), per mappare la struttura di ogni termine X ∈ L(S) all'algebra degli insiemi ℬ ≅ A(S). Assumiamo che: (a) X∈ L(S); (b) una partizione di Ed in sottoinsiemi join-irriducibili ∈ () è noto; (c) una mappatura uno a uno Ak ↦ pk tra atomi e i punti interni sono dati; (d) è disponibile un oracolo SMC, ovvero un test di classificazione dell'appartenenza a un insieme (Tilove, 1980). Un approccio ingenuo a SMC, dove ogni singolo punto pk (all'interno di un atomo Ak ∈ L(H)) è testato contro tutti i termini solidi di input X∈ H(S) possono essere calcolati in tempo quadratico O(NM), dove N è il numero di atomi e M è il numero di termini solidi di input X. Un'efficiente procedura O(N logM) è qui stabilita usando due (cioè, d − 1) alberi di intervallo unidimensionali per la scomposizione { Ak } di E3 , per eseguire il test SMC solo contro il termini nel sottoinsieme I(pk) ⊆ U3 , le cui scatole di contenimento intersecano un raggio dal punto di prova.

Algoritmo 10 (Genera e verifica). Pertanto, la struttura del termine X nell'algebra finita L(S) può essere calcolata utilizzando la procedura di generazione e verifica. Tale test SMC è semplice e non comporta la risoluzione di "on-on ambiguità” Tilove (1980), per la scelta di un punto interno pk in ogni atomo di algebra Ak . Set Membership Classification(SMC) tramite un test di contenimento del punto-poliedro.



Nella nostra attuale implementazione in 3D il test SMC viene eseguito intersecando un raggio da pk con i piani contenenti le 2 celle di atomi in I(pk ), e verificando il contenimento del punto-poligono in questi piani (tramite mappe nel sottospazio z=0). In sintesi, scomponiamo Ed in elementi irriducibili Ak di algebra L(S), e rappresentano ogni Ak con un punto pk. Per il teorema della curva di Jordan, un numero dispari di intersezione del raggio per pk con 2 celle al contorno di X produce una risposta oracolo sull'istruzione query pk ∈ X, e quindi Ak ⊆ X. Un numero pari dà il contrario.

3.6.4. Rappresentazione binaria di termini booleani

Costruiamo una rappresentazione di ogni termine X di un'espressione booleana solida, come un sottoinsieme di U3(base di 3 catene). Ricorda che, per costruzione, le partizioni U3 sono E3 e gli oggetti solidi di input.

Esempio 3.6.1 (Disposizione degli spazi dall'albero dell'assieme). Consideriamo l'assieme costruito mettendo insieme tre istanze di cubo unitario, opportunamente ruotate e traslate. La semantica di Lar.Struct() è simile a quella di PHIGS strutture (Kasper e Arns, 1993; Paoluzzi, 2003): The Lar constructor cuboidGrid of grids of cubes with “shape” [m, n, p], restituisce (con il parametro true opzionale) l'intera raccolta di p-celle, 0 ≤ p ≤ 3 in array di array di vertici indici VV,EV,FV,CV. Solo V,FV,EV (vertici, facce, bordi) sono effettivamente necessari per la generazione booleana.

assembly = Lar.Struct() è una linearizzazione del DFS (Depth First Search) dell'albero; Lar.arrangement() funzione applicato all'assieme restituisce la (geometria, topologia) della partizione spaziale 3D generata da esso. La geometria è data da la matrice di incorporamento W di tutti i vertici (0-celle), e la topologia dalle matrici sparse CF, FE, EV, cioè da δ2, δ1 , δ0 di complesso di catene che descrive la disposizione A(assieme). Vedere la definizione 3.6.3 (Complesso geometrico LAR).

Definizione 3.6.1 (vettori di bit). Un sottoinsieme Y di X può essere identificato con una famiglia indicizzata di bit con insieme indice X, con il bit indicizzato da x ∈ X essendo 1 o 0 a seconda che x∈Y oppure no. L'algebra booleana ℬ = P(X) dell'insieme di potenze di X può essere definito equivalentemente come l'insieme non vuoto di vettori di bit, tutti di lunghezza n= #X, con # ℬ = 2n .

Algoritmo 11 (Trasformazione di 3-catene in array di bit). Per tradurre una formula CSG solida in linguaggio macchina è sufficiente, una volta calcolata la matrice [3], e quindi la base U3, per ogni unità 3-catena uk∈U3 , per verificare l'appartenenza all'insieme di un singolo punto interno pk∈ uk , controllando se pk ∈X. In caso affermativo il k-esimo bit di coordinata vettore [X] ∈ {0, 1}n ≅ P (U3 ) è impostato su vero.

Esempio 3.6.2 (matrice booleana). Il valore dell'array di tipo Bool restituito nella variabile boolmatrix contiene per colonna i risultati di efficienti prove di contenimento solido punto-solido (SMC) per 3 celle atomiche (righe), rispetto ai termini di 3 partizioni: le 3 celle esterne e ciascuna C1 ,C2 ,C3 istanza cubo (colonne matrice), opportunamente mappata al mondo coordinate per valutazione struct.

Nota (termini booleani). I nostri oggetti C1 ,C2 ,C3 vengono estratte dalle colonne boolmatrix nelle variabili A,B,C, quindi descrivendo come ognuno è partizionato da (ordinato) 3 celle in 3 . L'intero spazio è dato da X = Ω ∪ A ∪ B ∪ C.

3.6.5. Risoluzione bit a bit di espressioni algebriche insiemi

Quando i solidi iniziali di input sono stati mappati su array di booleani, ora chiamati A, B, C, qualsiasi espressione del loro l'algebra booleana finita è valutata da operatori logici, che operano confrontando corrispondenti bit di variabili.

Definizione 3.6.2 (Operatori bit a bit). In particolare, il linguaggio Julia offre operatori logici bit a bit e (&), o (|), xor (⊻) e complemento (!), nonché un meccanismo di punti per applicare qualsiasi funzione agli array in base agli elementi. Quindi, noi possiamo scrivere espressioni come (A .& B) o (A .| B) che vengono valutate bit per bit e restituiscono il risultato in un nuovo BitArray vettore. Notiamo che questi operatori possono essere usati anche in forma prefissa e variadica. Quindi, gli operatori bit a bit possono essere applicati contemporaneamente a un numero finito di variabili.

Esempio 3.6.3 (Formule booleane). Seguono alcuni esempi. L'ultima espressione è l'intersezione del primo termine A con complemento di altri termini B e C, con A, B, C dell'Esempio 3.6.2, dando la differenza di insieme (A⧵B)⧵C. La variabile AminBminC contiene il risultato della differenza impostata indicata con &(A), B, C), mappata nel modello in Figura 1.

Esempio 3.6.4 (Algebra solida 3D). Calcoliamo la disposizione di E3 prodotta da 8 cubi unitari concentrici correvano ruotati intorno all'origine. L’insieme di dati di input è composto da 6 × 8 = 48 2 celle quadrate. La configurazione dell'ingresso è vicino al caso peggiore O(n2 ) per le operazioni booleane, poiché ogni faccia è intersecata da molte altre facce di input. Nel immagini sotto vediamo: (a) le facce frammentate dopo la scissione a 2 celle; (b) le 3 celle solide assemblate per dare il solido unione; (c) l'insieme esploso degli atomi dell'algebra booleana associati a tale disposizione; (d) un singolo (molto complesso) 3 cellule evidenziate. L'atomo centrale più grande è l'intersezione di tutti i cubi ed è un'approssimazione a 3 sfere.

Osservazione (robustezza topologica). La disposizione di E3 mostrato nell'Esempio 3.6.4 è un 3-complesso con 2208 vertici, 5968 bordi, 5360 facce e 1600 celle solide. La caratteristica di Eulero χ è χ 0−χ 1+χ 2−χ 3 = 2208 − 5968 + 5360 − 1600 = 0. Questo conteggio include le 3 celle esterne (illimitate). Ricordiamo che lo -spazio euclideo è topologicamente equivalente alla d-sfera meno un punto. la caratteristica di Eulero della d-sfera è χ = 1 + (−1)d = 2 o 0, per dimensione spaziale d pari o dispari . Vale la pena considerare la forma complessa di alcune 3 celle e 2 celle. Questi sono gestiti dalla rappresentazione LAR con la stessa semplicità dei triangoli o dei tetraedri.

Esempio 3.6.5 (esempio CSG — 2/2). In questo esempio completiamo il calcolo dello spazio booleano CSG indotto dalla partizione spaziale generata dalla formula mostrata nell'Esempio 2.3.1, e calcolare diverse altre forme su la stessa algebra booleana ℬ, utilizzando il nuovo metodo proposto in questo articolo. Ricordando che abbiamo cinque termini di input S= {X1 , X2 , X3 , Y, Z}, corrispondenti a tre istanze del cilindro , un cubo e una sfera. La disposizione di E3 che generano è A(S), mostrata di seguito, insieme a diversi valutati formule di questa algebra booleana (vedi Sezione 3.7). Il valore restituito da una macro @CSG è di tipo GeoComplex, ovvero a complesso geometrico (vedi Definizione 3.6.3). Vale la pena notare che due solidi risultanti vengono memorizzati come valutati Valori GeoComplex all'interno delle variabili A e B e che i simboli di tali variabili possono essere utilizzati in altre macro @CSG espressioni. Dall'alto verso il basso e da sinistra a destra, mostriamo:

@CSG (+,X1 ,X2 ,X3 ,Y,Z); A = @CSG (-,(\*,Y,Z),X1 ,X2 ,X3 ); @CSG (+,A,X1 ); @CSG (+,X1,X2); @CSG (+,A,(\*,B,Z)); B = @CSG (+,X1,X2,X3)); @CSG X1 ; @CSG (+,(\*,Y,Z),A); @CSG (+ B, Y, Z).

Ci sono 40 atomi nell'algebra ℬ mostrata qui e 240 ≃ 1012 termini con struttura diversa. Naturalmente, la rappresentazione delle coordinate di ciascun atomo è un bitArray di 40 elementi con un solo bit a 1 (vero) e tutti gli altri a 0 (falso). Chiaramente, per il calcolo di formule CSG con algebre più grandi, si possono usare vettori sparsi.

3.6.6. Calcolo del confine Nella maggior parte dei casi, l'ambiente computazionale geometrico di destinazione è in grado di visualizzare, più in generale di gestire, un modello solido solo utilizzando una rappresentazione del contorno, tipicamente una triangolazione. È facile ottenere tale rappresentazione moltiplicando la matrice dell'operatore 3-boundary 3∶ C3 → C2 volte il vettore coordinate nello spazio C3 dell’ espressione solida, calcolata come un termine binario della nostra algebra degli insiemi. Una volta ottenuto in questo modo il vettore di coordinate con segno del confine dell'oggetto solido, cioè la 2-catena delle sue 2-celle orientate (facce), queste devono essere raccolte per colonne in una “matrice di facce” sparsa e tradotta nella corrispondente matrice di 1-cicli di spigoli orientati, mediante moltiplicazioni giuste di [2] volte la matrice della faccia. I poligoni di contorno generati saranno infine triangolati e resi dal hardware grafico o esportati in formati di file grafici standard o in altri formati necessari per le applicazioni. Definizione 3.6.3 (Complesso geometrico LAR). Vale la pena notare che, per mostrare una triangolazione di facce di confine nella loro corretta posizione nello spazio, l'intera informazione richiesta (geometria + topologia) è contenuta all'interno del Complesso Geometrico LAR (GC):

μ ∶ C0 → E3 , (δ2, δ1, δ0) ≡ (geometria, topologia) = (W, (CF, FE, EV)).

Un GC permette di trasformare il confine (possibilmente non connesso) 2-ciclo di un risultato booleano (vedi l'esempio sotto) in una B-rep completa del risultato solido. Nota che le coppie ordinate di lettere da V,E,F,C, corrispondono al coconfine sequenza Vertici→Bordi→Facce→Celle nella colonna→Ordine di riga delle mappe matriciali degli operatori.

Osservazione (rappresentazione a catena). Ricordiamo che CF (ovvero Facce → Celle) è la matrice sparsa dell'operatore co-confine δ2∶ C2 → C3 , così che abbiamo [3 ] = [δ2 ]’ , che in Julia è CF’. Il valore di AminBminC = .&(A,.!B,.!C) dato nell'Esempio 3.6.6, è la rappresentazione C2 del confine orientato del 2-ciclo di differenza solida, visualizzato in Figura 1.

Nota (dalle catene solide alle ripetizioni B). Naturalmente, i 14 elementi diversi da zero nell'array di confine (Esempio 3.6.6) della variabile AminBminC, corrispondono alle 2-celle di confine orientate del risultato solido (vedi Esempio 3.6.6). Ciascuna di loro viene trasformata in un 1 ciclo possibilmente non connesso dalla matrice sparsa FE', cioè dalla matrice 2-confine[2] = [δ1]’ . Infine, ogni ciclo 1 viene trasformato in una o più sequenze cicliche di celle 0, utilizzando la matrice EV', cioè, usando [1] = [δ0] ‘. Gli indici delle 0 celle sono ordinati ciclicamente e utilizzati per generare sequenze di punti 3D tramite la matrice di immersione W, che fornisce le coordinate dei vertici per colonna. Quest'ultimo passaggio fornisce l'input ordinato per la triangolazione del viso utilizzando un algoritmo CDT (Constrained Delaunay Triangulation) (Shewchuk, 2002) in 2D.

Proprietà 3.6.2 (Spazio di stoccaggio del Complesso Geometrico LAR). La topologia di un complesso LAR 3 è completamente rappresentata dagli operatori δ0, δ1, δ2 , cioè dagli array sparsi (EV,FE,CF), fornendo le incidenze tra vertici, bordi e volti, sia per B-reps che per rappresentazioni decompositive. Se viene utilizzata una rappresentazione del confine, la memoria LAR è 3/4 la rappresentazione del bordo alato di Baumgart (1972), spesso usata come confronto di archiviazione per la modellazione solida, e molto vicino (3 × 2E, vedi Woo (1985)) a Muller e Preparata (1978), largamente usato in Computational Geometria.

Esempio 3.6.6. Confine di espressione solida. La variabile AminBminC contiene la rappresentazione logica della Espressione booleana (A⧵B)⧵C, specifica della particolare algebra generata dai termini dell'Esempio 3.6.1:

1 julia> difference= Int8.(AminBminC)

2 8-element Array{Int8,1}:

3 [0, 0, 0, 0, 1, 0, 0, 0]

4

5 julia> boundary= CF’ \* difference

6 47-element Array{Int8,1}:

7 [1, 0, -1, 0, 0, 0, -1, 0, 1, 0, 1, 0, 0, 0, 1, -1, 0, -1, 0, -1, 0, 0 , 1, 0, 0, 0, -1, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 1, 0, 0, 0, 0, -1, 0, 0, 0, 0].

Il valore della variabile AminBminC viene convertito nella differenza dell'array binario, la rappresentazione delle coordinate di una 3-catena , dal costruttore vettorizzato “Int8.”, per calcolare l'oggetto di confine dell'espressione booleana che è mostrato in Figura 1, moltiplicando per la matrice di confine [3] ≡ CF’ Infine, notiamo che in notazione a catena è possibile scrivere la seguente espressione per il confine orientato del solido ternario Differenza booleana. La riduzione effettiva al confine triangolato si ottiene utilizzando le matrici sparse FE' ed EV', e un algoritmo di triangolazione di Delaunay vincolato (CDT).

c ↦ fA∖B∖C = f1 − f3 − f7 + f9 + f11 + f15 − f16 − f18 − f20 + f23 – f27 − f33 +f 38 − f43

3.7. DSL booleano In questa sezione discutiamo il progetto Domain Specific Language (DSL) sul Constructive Solid introdotto dall'algebra della geometria (CSG), come piccolo insieme di macro di Julia, è attualmente in fase di sviluppo.

Definizione 3.7.1 (Forma funzionale). Sia ⟨A; ⊗1 ,…, ⊗k≻ essere un'algebra generata da H= {ℎ1 , ..., ℎm}. Un’espressione sintattica costruita come una sequenza valida di operazioni ⊗i su n variabili con xi che denota elementi di A si chiama forma funzionale Φ sull'algebra A. Nota che Φ(x1 , ..., xn ) non è una funzione (non è un insieme di coppie ordinate) ma definisce una funzione di n argomenti Φ ∶ An→ A. Osservazione (processo di valutazione). Usiamo lettere greche maiuscole come Φ, Π, ecc. per denotare forme e denotare funzioni su algebre con lettere greche minuscole. La distinzione tra forme e funzioni è importante. Forme e funzioni su un'algebra sono formalmente correlati da un processo di valutazione, assegnando valori alle variabili nella forma e calcolando il valore risultante dell'espressione. Questa relazione è espressa scrivendo Φ (x1 , ..., xn ) = |Φ(x1 , ..., xn)|.

Proprietà 3.7.1 (Processo di valutazione). Se A è un'algebra finita, esiste un numero finito di funzioni distinte su A , ma un numero infinito di forme distinte, ad esempio fortemente ridondanti. Se S ∈ A è un elemento dell'algebra generato da H, esiste una forma Φ su A tale che: S = |Φ(H)|. Diciamo allora che S è descrivibile in A da H. In generale, Φ(H) non è unico, ma |Φ(H)| è unica per definizione.

Definizione 3.7.2 (lingua specifica del dominio). Il Domain Specific Language (DSL) proposto in questo documento per la nostra algebra CSG è molto semplice. Le sue formule ben formate sono costituite da identificatori di variabili Julia, simboli di operazione booleana e parentesi tonde. Un'espressione CSG è codificata come una macro Julia @CSG, per fornire un'interfaccia più semplice per creare Oggetti Julia con struttura complicata. L'elenco dei modelli di oggetti associati alle variabili viene estratto per attraversamento del l'espressione @CSG. Ogni oggetto deve essere localizzato e orientato in coordinate globali usando il tipo di dati Lar.Struct, permettendo di combinare insiemi gerarchici (complessi cellulari in coordinate locali) e trasformazioni affini. L’applicazione della funzione Lar.struct2lar() valuta un oggetto geometrico di tipo Lar.Struct, generando la tripla (V,FV,EV), ovvero i modelli minimi LAR (DiCarlo et al., 2014) utilizzati dal pacchetto CSG.jl.

Algoritmo 12 (Processo di valutazione). La valutazione di una forma funzionale booleana 3D @CSG <expr>, in un modello LAR valutato (V,(CF,FE,EV)), costituito da un array V 3 × n e da una tripla (CF,FE, EV) di array sparsi, fornendo rispettivamente l'immersione geometrica C0→ E3 , e un complesso a catena δ2, δ1, δ0 è composto da diversi compiti:

1. Valutazione dei dati e delle forme funzionali. **Input:** un singolo blocco macro @CSG, possibilmente delimitato dalla clausola inizio...fine, inclusi termini solidi e (possibilmente) clausole @CSG del componente. **Output:** analizza l'albero del composto @CSG espressione; tupla ordinata di nomi di variabili associate agli oggetti di input in un singolo frame di coordinate: (A1 ,…,Am).

2. Calcolo della disposizione degli spazi. **Input:** tupla di termini solidi in coordinate mondiali: (A1 ,…,Am). **Output:** matrici di confine sparse (FC,EF,VE) come [3], [2], [1], e (nuova) matrice di vertici W. Le colonne di [3] sono uno a uno con gli atomi che generano l'algebra @CSG.

3. Valutazione dei termini booleani**. Input:** la rappresentazione al contorno degli atomi dell'algebra CSG, generata dalla catena complessi (W,(FC,EF,VE)) e i termini di input (A1 ,…,Am). **Output:** array nxm B = Array{Bool,2} la cui colonna Bj fornisce l'array di bit che rappresenta l'Aj termine del modulo di input algebra @CSG. 4. Calcolo bit a bit della funzione booleana. **Input:** l'array logico B; l'albero di analisi dell'espressione @CSG di primo livello. **Output:** un singolo BitArray di lunghezza , come il numero di atomi booleani. L'1s in questo array denota l'atomico 3-catene la cui unione disgiunta produce il risultato booleano, ovvero il valore della forma @CSG valutata. 5. Valutazione del confine dell'output 3-catena. **Input:** la rappresentazione delle coordinate (binarie) a 3 catene del valore booleano risultato; il complesso di catene W,(FC,EF,VE). **Output:** la rappresentazione del bordo del risultato booleano, possibilmente esportati in un formato di grafica vettoriale standard, come, ad esempio, i formati di file .OBY, .PLY o .DAE (Digital Asset Scambio — COLLADA XML). Il nostro prototipo software può attualmente esportare in 2D e 3D .OBY.

Definizione 3.7.3 (forma booleana). Clausola (espressione formata da una raccolta finita di letterali) che corrisponde ad altre clausole e/o modelli LAR, abbinando combinazioni booleane di altre clausole. È costruito utilizzando uno o più @CSG macro, ognuna con un'occorrenza "Struct" digitata. La semantica di una Struct, comprendente nativamente solo “aggregazione” e "trasformazione affine", si arricchisce di simboli n-ari di unione, intersezione, differenza e complemento. A, B, C,… possono essere modelli LAR o valori letterali o valori Struct. Abbiamo: :(+, A,B,C,...)::= Struct (:union, [A,B,C,...]) :(\*, A,B,C,...)::= Struct (:interseca, [A,B,C,...]) :(-, A,B,C,...)::= Struttura (:diff, [A,B,C,...])

4. Esempi computazionali Questa sezione fornisce al lettore alcuni esempi di stile di programmazione di modellazione solida con Julia e i suoi scarsi e matrici non sparse. Crediamo che dare uno sguardo ad alcuni semplici esempi concreti sia utile per capire il nostro approccio computazionale e i suoi possibili sviluppi. Le sezioni 4.1 e 4.2 mirano a mostrare la sequenza di spazi e trasformazioni che alla fine producono un modello solido quando si valuta un'espressione booleana attraverso una funzione applicazione Lar.bool3d (assieme). Mostriamo nella Sezione 4.3 che la proprietà di esattezza (2= 0) di qualsiasi catena complesso può essere utilizzato per verificare l'accuratezza dei calcoli. La caratteristica di Eulero del modello solido di unione generato nell'Esempio 3.6.1 è calcolato passo per passo nella Sezione 4.4, dove usiamo le mappe a catena 3 , 21 per ottenere gli insiemi (e numeri) di facce, spigoli e vertici appartenenti al confine del solido di unione, che è una 2-varietà chiusa di genere zero (vedi Figura 2). Come abbiamo detto prima, sono in fase di progettazione un'API specifica e un mini DSL per le espressioni CSG.

4.1. Unione variadica Per calcolare l'unione di tre istanze affinemente trasformate del cubo unitario, consideriamo l'assieme espressione data nell'Esempio 3.6.1. Per prima cosa otteniamo la partizione spaziale E3 e la struttura del termine generata dall'assieme tramite la funzione Lar.bool3d; quindi combiniamo gli array logici A, B e C, costruendo il valore di tipo BitArray per la variabile union, che memorizza la rappresentazione logica della specifica 3-catena. Notiamo che l’operatore bit a bit "o" (“|”) viene applicato in modo vettorizzato agli array, inserendo un carattere punto:

1 julia> W, (EV, FE, CF), boolmatrix = Lar.bool3d(assembly);

2 julia> A,B,C = matricebool[:,2],matricebool[:,3],matricebool[:,4]

3 julia> unione = .|(A, B, C);

4 julia> @show sindacato;

5

6 union = Bool[falso, vero, vero, vero, vero, vero, vero, vero]

Infine, le facce a 2 cicli di contorno sono generate dalla moltiplicazione della matrice sparsa [3] (cioè CF’) volte il unione binaria convertita. La mappatura Bool → {0, 1} viene applicata tramite un'applicazione vettorizzata del costruttore Int8:

1 julia> face = CF’ \* Int8.(unione);

2 julia> @mostra faces;

3

4 faces = [1, 0, -1, 0, 0, 0, -1, 0, 1, 0, 1, 0, 0, 0, 1, -1, 0, -1, 0, 0, 1, -1, 0, -1, 1, 5 0, 0, 0, 1, 1, 0, -1, 0, 0, 1, 0, -1, 0, 0, 0, -1, 1, 0, 1, 0, 0, -1].

Con fk ∈ U2 , dove U2 è la base dello spazio delle catene C2 generato da A (assieme), possiamo scrivere in notazione a catena:

faces ↦ fAUBUC =f1 − f3 − f7 + f9 + f11 + f15 − f16 − f18 + f21 − f22 − f24 +f 25 + f29 +f 30 − f32 + f35 −f 37 − f41 + f42 +f 44 −f 47.

La rappresentazione del contorno, cioè il sottoinsieme delle facce orientate del contorno, è data dal 2-ciclo nell'equazione precedente. Essi si trasformano in una triangolazione di confine, necessaria per la visualizzazione grafica, mediante l'uso appropriato di [2 ] = FE' e [1] = EV'.

4.2. Significato di matrici Per definizione, la matrice [2 ] = FE’ = EF contiene per colonne la base U2 espressa come sequenza ordinata di Vettori a 1 ciclo. I valori con segno forniscono un ordinamento ciclico di 1 ciclo di bordi, facilmente estendibile a dimensioni superiori. Con eh ∈U1 , noi abbiamo:

1 FE'[:, 1] ↦ f1 = e1 -e2 +e4 -e5 +e6 -e7 +e8

2 FE'[:, 2] ↦ f2 = e3 +e7 -e8

3 ... ... ...

4 FE'[:,47] ↦ f47 = e64 -e69 +e77 -e81 -e86 +e88.

Ogni 1 cella è mappata su una coppia ordinata di 0-celle vj∈ U0 , tramite la matrice di confine [1] = [δ0 ]’= EV':

1 EV'[:, 1] ↦ e1 = v2 - v1

2 EV'[:, 2] ↦ e2 = v6 - v3

3 ... ... ...

4 EV'[:,49] e49 = v29 - v26.

L'incastonatura geometrica in E3 del modello AUBUC generato nella Sezione 4.1 è fornito dalla coordinata matrice W ∈ ℝ3, con 3 righe e 43 colonne. La matrice di coordinate V che incorpora il modello di assieme iniziale, consistente di tre cubi non (ancora) intersecati, ha invece dimensione 3 × 24.

4.3. Controlli di correttezza Un approccio computazionale basato su complessi a catena offre strumenti unici per verificare l'accuratezza dei calcoli, che sono corretti per costruzione, poiché entrambi 2= 0 e δ2 = 0, quando applicato a qualsiasi catena. In parole, ogni confine è un ciclo, o equivalentemente: il complesso della catena è esatto. Inoltre, sappiamo che il confine di ogni solido è a possibilmente superficie chiusa non collegata, quindi un ciclo a 2 tempi. Ciò vale se e solo se la costruzione della matrice [] è fatto correttamente. Nell'esempio seguente, partiamo dalla catena di facce di contorno della Sezione 4.1.

1 julia> pairs = [(f,sign) for (f,sign) in enumerate(copCF’ \* Int8.(union)) if sign != 0];

2 julia> facce = map(prod, pairs);

3 julia> @show faces;

4 faces = [1,-3,-7,9,11,15,-16,-18,21,-22,-24,25,29,30,-32,35,-37,-41,42 ,44,-47].

Otteniamo la seguente 2-catena come 2-ciclo di confine. La cardinalità dell'array delle facce è χ2= 21.

facce ↦ fAUBUC =f1 − f3 − f7 + f9 +f 11 +f 15 − f16 −f18 + f21 − f22 − f24 + f25 +f29 + f30 −f 32 + f35 −f 37 − f41 +f 42 +f 44 −f 47.

Il sottoinsieme degli archi sul bordo del solido di unione è calcolato: (a) trasformando l'array delle facce degli indici con segno di 2 celle nella rappresentazione COORD (rows,cols,vals) di una matrice sparsa (Cimrman, 2015); (b) in possesso di una singola faccia di confine (2 celle) per colonna in facemat; (c) moltiplicando questa matrice per l’operatore[2 ] , ottenendo così la nuova matrice sparsa edge4face, contenente una faccia 1 ciclo per colonna; e, infine, (d) estraendo solo l'istanza positiva dei bordi di confine appartenenti alle facce di confine. Il numero di bordi di confine χ1= 57 è la metà dei termini diversi da zero nel vettore sparso edge4face. L'altra metà ha il segno opposto, per cui la somma totale è zero:

1 julia> nonzeros = hcat([[abs.(face),k,segno(face)] for (k,face) in enumerate(faces)]...);

2 julia> facemat = sparse([nonzeros[k,:] for k=1:size(nonzeros,1)]...);

3 julia> edge4face = copFE’ \* facemat;

4 julia> rows,cols,vals = findnz(edges4face);

1 julia> edges = [e\*sign for (e,sign) in zip(row,vals) if sign==1];

2 julia> @show edges;

3

4 edges = [1,4,6,8,10,14,18,20,9,23,26,27,2,34,30,36,38,5,29,24,43,15,42, 28,47,51,53,54,21,52,57, 5 48,61,62,64,50,59,66,55,58,40,63,68,77,79,80,35,78,83,72,88,7,74,87,69,81 ,86].

Notiamo ancora che, senza filtrare i termini di segno negativo, otterremmo un array di archi di segno indici sommando a zero, secondo il vincolo 2= 0. Ciò attesta l'esattezza dei calcoli.

* 1. caratteristica di Eulero La caratteristica di Eulero del confine connesso di un solido di genere g in E3 è definito come:

χ (g) = χ0 – χ1 + χ2= 2 – 2g, dove χ0 ,χ1 ,χ2sono, rispettivamente, uguali al numero di vertici, spigoli e facce sul contorno del solido. In nel nostro caso, il numero di facce al contorno è χ2= 21. Gli indici degli archi forniti nella Sezione 4.3 determinano la 1-catena eAUBUC (o, più precisamente, l'1-ciclo non indipendente generato dai 2-cicli indipendenti delle facce di confine). La cardinalità dell'array degli archi fornisce χ1 = 57. Si noti che, per contare gli archi, consideriamo solo le istanze positive di 1-celle, poiché appaiono in coppia (positivo e negativo) in un chiuso e 2-complesso cellulare coerentemente orientato.

edges ↦ eAUBUC = e1 + e4 + e6 +e8 + e10 + e14 + e18 +e20 +e9 +e23 +e26 +e27 + e2 +e34 + e30 + e36 + e38 + e5 +e29 +e24 + e43 + e15 + e42 + e28 + e47 + e51 + e53 + e54 + e21 + e52 + e57 + e48 + e61 + e62 + e64 + e50 + e59 + e66 + e55 + e68 + e77 + e79 + e58 + e40 + e63 +e80 +e35 +e 78 + e83 + e72 + e88 + e7 + e74 + e87 + e69 + e81 + e86.

Lo script finale calcola la 0-catena vAUBUC dei vertici sul confine del solido di unione, con cardinalità χ0=38.

1 julia> diverso da zero = hcat([[abs(e),k,sign(e)] for (k,e) in enumerate(edges)]...);

2 julia> edgemat = sparse([nonzeros[k,:] for k=1:size(nonzeros,1)]...);

3 julia> verts = sort(collect(Set(findnz(copEV’ \* edgemat)[1])));

4 julia> @show verts;

5 vertS= [1,2,3,4,5,6,7,8,9,10,12,15,17,18,19,20,21,24,25,26,27,30,31,32,33,34,35,36,37,38,39,41,44,45,46,47,48,49].

In notazione a catena:

verts↦ vAUBUC= v1 + v2 + v3 + v4 + v5 + v6 + v7 + v8 + v9 + v10 + v12 + v15 + v17 + v18 +v 19 +v 20 +v 21 +v 24 +v 25 + v26 +v 27 + v30 + v31 + v32 + v33 +v 34 +v 35 + v36 +v 37 + v38 + v39 + v41 + v44 + v45 +v 46 +v 47 + v48 + v49.

Il modello solido di unione generato ha genere topologico g = 0 (vedi Figura 2). Pertanto, il test di correttezza fornito verificando la caratteristica di Eulero tramite le eq. (5), (6) e (7), dà la risposta corretta: χ(unione) = χ0 - χ1+ χ2 = 38 − 57 + 21 = 2.

5.Discussione sui risultati L'approccio proposto in questo articolo è nuovo sotto diversi aspetti. Prima di tutto, abbiamo introdotto qui il primo metodo di geometria computazionale solida, a conoscenza degli autori, per assemblare i termini di una forma CSG utilizzando gli atomi di Algebra booleana, identificata con le 3-catene di base di una partizione dello spazio di immersione, a sua volta data dalle colonne della matrice [3] nello spazio delle catene. Inoltre, abbiamo iniziato a impostare la struttura dei termini della forma booleana come array binari, ovvero vettori di coordinate nello spazio C3 e combinandoli con operatori bit a bit nativi. Inoltre, con il unica eccezione di d-tree e interval-tree per l'accelerazione delle query geometriche, il nostro approccio non utilizza metodi standard di geometria computazionale - in particolare, di sottocampo di modellazione solida - che si basano su metodi altamente specializzati di strutture dati, che spesso sono molto complesse e richiedono l'implementazione di algoritmi molto complicati. Utilizziamo invece strumenti e metodi di base dell'algebra lineare e della topologia algebrica, ovvero matrici (sparse) di operatori e moltiplicazione di matrici semianelli (Kepner et al., 2016), più filtraggio. Quindi, l’accuratezza del calcolo topologico è garantita dalla costruzione, poiché le matrici degli operatori e le basi della catena, così come qualsiasi catena, soddisfano i vincoli (gradati) 2= 0 e δ2 = 0, facilmente verificabili. Inoltre, abbiamo mostrato che spazio e tempo complessità sono paragonabili a quelle dei metodi tradizionali, ed è ragionevole aspettarsi che il nostro approccio possa essere esteso a dimensioni generiche e/o implementato su motori di calcolo altamente paralleli, utilizzando piuttosto standard e librerie di calcolo molto generali, ad esempio GraphBLAS e kernel GPU. In questo caso è prevista una buona accelerazione, dato l'alto livello di possibile parallelizzazione dei nostri algoritmi. In questo articolo abbiamo anche dato una nuova caratterizzazione del processo computazionale per la valutazione delle espressioni CSG di qualsiasi profondità e complessità. I metodi di valutazione tradizionali richiedono un attraversamento DFS post-ordine dell'albero delle espressioni, e il calcolo sequenziale di ogni operazione booleana incontrata sui nodi, fino a quando non viene valutata l'operazione radice. È noto che un tale processo manca di robustezza e accumula errori numerici, che alla fine modificano la topologia locale e arrestare le applicazioni in caso di errore. Quindi, le rappresentazioni di confine intermedie devono essere generata e curata con cura, prima di continuare la traversata. Con il nostro approccio, la valutazione di un'espressione CSG di qualsiasi complessità viene eseguita con un diverso calcolo processi. Una struttura gerarchica di macro viene utilizzata sia per analizzare l'albero delle espressioni che per applicare trasformazioni affini a solidi primitivi, per ridimensionarli, ruotarli e tradurli nelle loro posizioni e attitudini finali (“mondo”). Tutto le loro 2 celle vengono così accumulate in un'unica collezione, e ciascuna di esse viene frammentata indipendentemente, generando una collezione di topologie locali che vengono fuse per congruenza di confine, utilizzando un'unica operazione di arrotondamento sui vertici. La partizione dello spazio globale è così generata e tutti i solidi di input sono classificati rispetto alle 3 celle di questa partizione, cioè, gli oggetti atomici di un'algebra booleana, con un test di appartenenza a un singolo punto per atomo. Infine, qualsiasi forma booleana di complessità arbitraria, a parità di variabili, possono essere rapidamente valutati mediante operazioni logiche vettorializzate bit per bit. Ultimo ma non meno importante, viene rimossa ogni distinzione tra geometrie molteplici e non molteplici, sia in 2D che in 3D, consentendo la miscelazione di B-reps e decomposizioni cellulari dell'interno dei solidi elementari. Le ripetizioni B possono essere sempre generato a fini di efficienza, tramite la moltiplicazione dell'operatore di confine, al fine di ridurre fortemente il numero di 2-celle da scomporre (possibilmente in parallelo), ma questo non è strettamente necessario. Questa flessibilità sui tipi di dati di input potrebbe essere utile, in particolare, per combinare superfici esterne con regolari griglie solide, ed è coerente con la generazione di strutture interne per rendere i modelli stampati in 3D più affidabili e facili da produrre.

6. Conclusioni e prospettive future Abbiamo introdotto un nuovo approccio al calcolo delle operazioni booleane tra modelli solidi. In particolare, abbiamo dimostrato che qualsiasi espressione booleana tra modelli solidi può essere valutata utilizzando l'algebra finita associata con l'insieme dei generatori indipendenti dello spazio di catena prodotto dalla disposizione dello spazio euclideo generato da una collezione di oggetti geometrici. L'approccio computazionale al disegno geometrico introdotto da questo articolo è abbastanza diverso dai metodi tradizionali di modellazione geometrica e geometria computazionale. Due prototipi di implementazione open source sono stati scritti in linguaggio Julia (Bezanson et al., 2017), principalmente utilizzando array sparsi. Al momento, possiamo valutare solo formule CSG con celle regolari chiuse. Una nuova implementazione per le algebre di chiusura è pianificata, utilizzando le funzionalità parallele e distribuite di Julia, che forniscono il supporto migliore per l'algebra lineare e i calcoli di matrici, incluso il nuovo standard GraphBLAS (Kepner et al., 2016). Riteniamo che, introducendo metodi lineari nella geometria solida, abbiamo in un certo senso solo scalfito la superficie del nuovo corpo di conoscenze attualmente oggetto di indagine mediante metodi di apprendimento automatico per la comprensione delle immagini, che usa anche tensori e algebra lineare. I metodi ML di nuova generazione richiedono tecniche formali e astratte per costruire e fusione di modelli solidi derivati ​​da immagini parziali. Abbiamo già fatto alcuni primi esperimenti della nostra topologia metodo algebrico nell'imaging medico 3D, combinando operatori di confine e coboundary con filtri, mirando a scoprire e rintracciare complesse strutture interne. Ci auguriamo che le tecniche introdotte da questo documento possano fornire alcuni strumenti di base per molte applicazioni in questo e in altri campi.